ENTRYPRO: FORM DESCRIPTION AND DATABASE INTERFACE SYSTEM

Jai Singh

A Major Report
in
The Department
of
Computer Science

Presented in partial fulfilment of the requirements
for the degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

August 1981

ENTRYPRO

A FORM DESCRIPTION & DATABASE INTERFACE SYSTEM

ABSTRACT

ENTRYPRO: FORM DESCRIPTION AND DATABASE INTERFACE SYSTEM

Jai Singh

ENTRYPRO is a form description and database interface system for on-line data entry type of applications. Using the instructions provided in ENTRYPRO language (DIL) a programmer/analyst can create a system of interlinking menus and form families. Optionally, the items in a form can be edited and validated by Item Processing Commands provided in ENTRYPRO. Several levels of access control are implemented on user classes. There may be upto 10 different user classes.

ENTRYPRO interfaces the forms to an on-line database. Since ENTRYPRO does not apply locks to the database for long durations, the data entries accessed for modification are re-read, compared and updated under a lock. This ensures that there are no "missing" updates to the database. If a compare test fails after re-read then the operator is warned and a new fetch on the database is requested.

ENTRYPRO provides an interface to user created programs or subprograms. It can execute any valid operating system command. The control returns to ENTRYPRO after termination of these processes.

ACKNOWLEDGEMENTS

I wish to express my gratitude to my advisor, Dr. Desai for his constant encouragement and guidance.

My gratitude is due to Lesters Foods Limited, Laval where the idea for this report originated, and who graciously afforded me the computing resources for implementing the package. This package is currently used in the company for developing of application systems.

I would like to thank my colleagues, Shiv Verma, Angelina Liberatore and Raymond Lai at Lesters for valuable suggestions during development and testing of the programs.

My many thanks to Sharon Jamison for providing with the secretarial support, and Joyce Gill for all the paper she mounted on line printer for endless listings.

TABLE OF CONTENTS

LIST OF FIGURES

1. ENTRYPRO SYSTEM DESCRIPTION

1.1 INTRODUCTION

ENTRYPRO is a software package which lets a programmer/analyst specify screen forms and describe their relationship to one or more datasets in the database. The data interchange is done interactively on a CRT, using the format defined by the analysts. The form selection is guided by having a main menu and submenus.

The form description and its database interface is achieved by building a specifications file by using DIL (Database Interface Language) instructions. The complete set of instructions is described using BNF in Appendix C. The forms in the system are related to one of the menus in the system, i.e. users can access one or more forms by making an appropriate selection from the menu when displayed by ENTRYPRO.

Database security is implemented by restricting access to the form. This is done by associating a set of user class numbers to a form. By restricting access to a submenu we can restrict the entire forms family associated with that submenu.

The present version of the system has been implemented on a HP3000 (III) minicomputer. The ENTRYPRO system programs are written in COBOL [1] and Systems Programming Language [2]. The forms are interfaced to the IMAGE database supported by HP3000 computer [3].

The system has been designed with a high degree of portability. Two of the modules are computer system dependent, i.e. database calls and I/O procedures for the interactive terminals (screen manager calls).

The internal organization of ENTRYPRO can be divided in two parts: the first part is a simple one pass LL(1) parser (SYSPACQ). It reduces the specification instructions to a pre-determined and complete tables containing necessary information. This stage generates listing and detects errors which are flagged by the parser. The second part of ENTRYPRO is used for actual execution of the tables generated by the parser. The tables generated by the first part are taken as inputs by the executive program (SYSPEXEC). The functions of the executive are as follows:

a) Displaying of MENUS and selection of forms.

b) Enforcing access restriction (security).

c) Displaying of forms and executing the form semantics.

d) Accessing and modifying database contents as related to the forms being accessed.

e) Ensuring database consistency and integrity during updates.

The tables produced by the part 1 can be listed by using a separate program (SYSPLIST) which produces the inner listing of the ERYPRO tables. This step is quite helpful in doing system testing during development and modification (at some later time).

The system level diagram of ENTRYPRO is given in Fig. 1 and 2.

1.2 DESCRIPTION OF DATABASE INTERFACE LANGUAGE (DIL)

The specification file contains a program written using DIL instructions. The complete grammar of DIL is given in Appendix A.

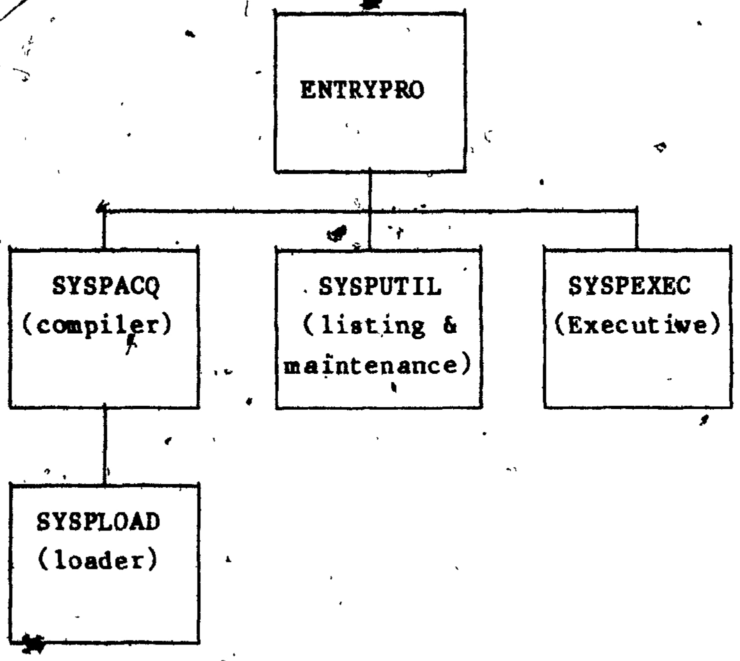
DIL instructions are format free however a new MENU or FORM must be started on a new line in column 1, this feature is used for error recovery.

The system analyst may create as many different specification files as needed. These files will be used for creating different systems. The first line of spec~~ification~~ file indicates the 4-character system code, which identifies the system to ENTRYPRO and SYSPAC database. A system is described by three different types of constructs; a) system environment, b) menus and c) form or form families.

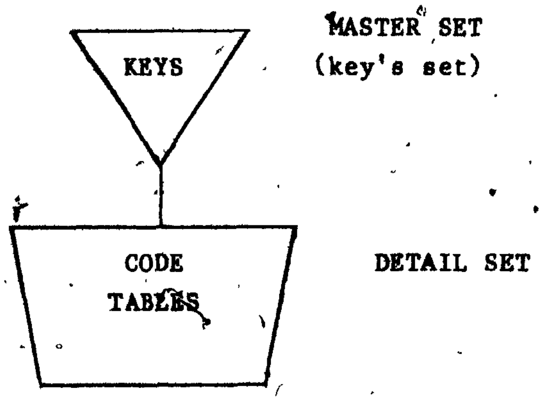
It is possible to create a system very quickly using only stub forms and other programs evoked via ENTRYPRO.

The form interaction with the terminal CRT is completely defined by ENTRYPRO and cannot be redefined by user programer. This is a limitation of ENTRYPRO. The ENTRYPRO interaction commands and messages are listed in Appendix D.

FIG. 1: ORGANIZATION OF ENTRYPRO



SYSPAC DATA BASE ORGANIZATION



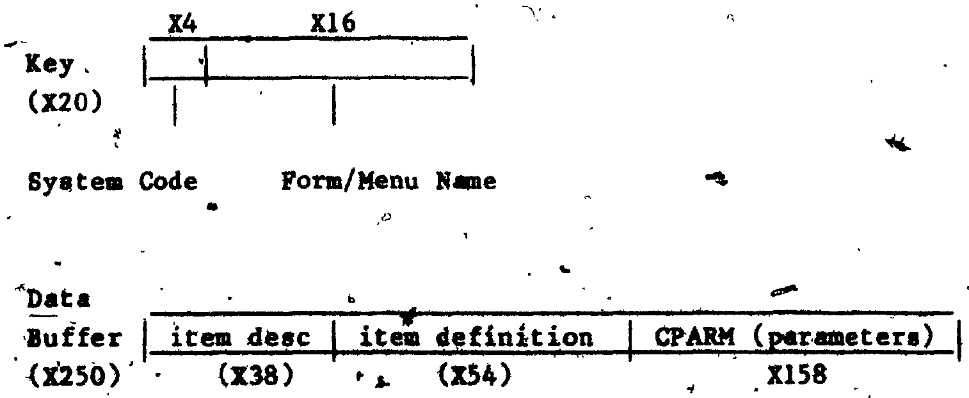
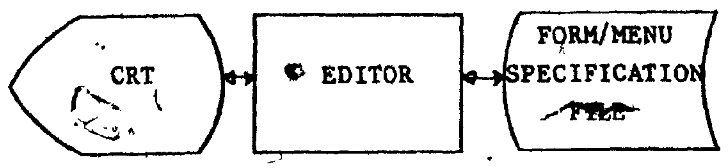
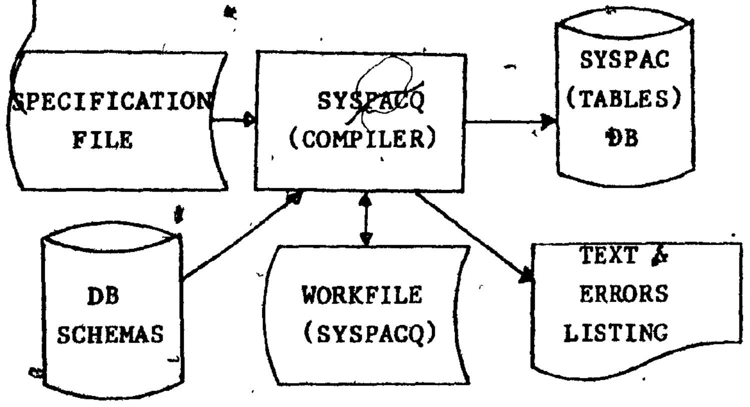
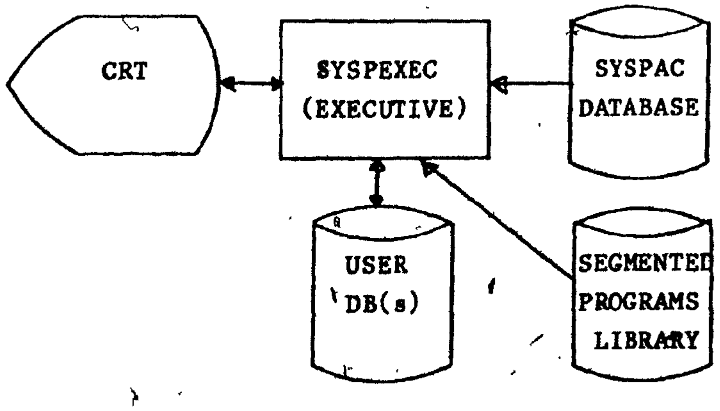


FIG. 2: FILE ORGANIZATION OF "ENTRYPRO" SYSTEM

	ACTION	MODE OF OPERATION
0		Interactive
1	 SPECIFICATION FILE SYSPACQ (COMPILER) SYSPAC (TABLES) DB DB SCHEMAS WORKFILE (SYSPACQ) TEXT & ERRORS LISTING	Batch or Background
2	 CRT SYSPEXEC (EXECUTIVE) SYSPAC DATABASE USER DB(s) SEGMENTED PROGRAMS LIBRARY	Interactive On-line

1.3 BUILDING THE SPECIFICATION FILE

The specification file is a simple editor type file. A specification file describes the inter-linking system of menus and forms (see Fig. 3). It can be built using text editor provided with the system software. The specification file consists of DIL instructions given in Appendix A.

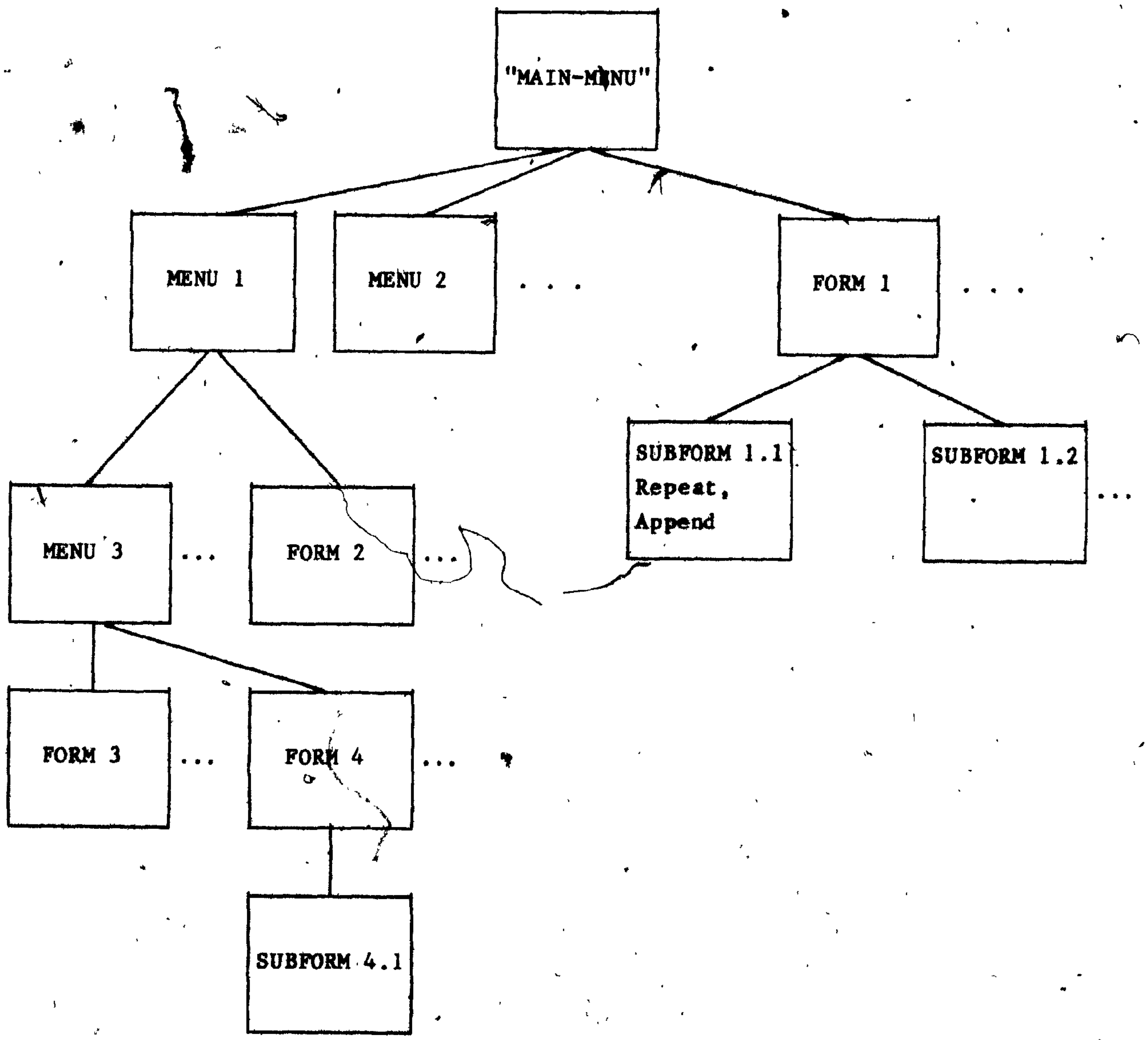
Of course, some thought must be given to the system design before one can start building the specification file. This system forces the programmer to consider only top down approach. Specification file without all the details can be tested anytime during development phase.

The specification file can be easily modified by using the editor subsystem, however it must be re-compiled after each modification.

The forms should be designed using a form description sheet. This can help the designer "see" the form layout before developing the ENTRYPRO specification file.

A system title is given to each file. The system title should not exceed 4 alphanumeric characters. The first menu in the file should be named "MAIN-MENU".

FIG. 3: MENU & FORM HIEARCHIES



2. COMPILATION AND TABLE GENERATION

2.1 INTERNAL ORGANIZATION - SYSPACQ

The first part of ENTRYPRO system consists of a compiler translator. The translator program is called SYSPACQ. It takes the specifications file created by user and reduces it to a set of tables. The tables are mostly of fixed format, which makes their interpretation and execution quite simple by another program called SYSPEXEC.

The SYSPACQ program is organized like a single pass compiler. It checks for syntax compliance and semantic errors. The errors when detected are flagged on the output. The complete list of possible errors are listed in Appendix B. The table generation (synonym to code generation) accompanies with the parsing actions of SYSPACQ.

The program itself has been written in COBOL and is therefore quite portable. To make it compatible with another system only the loader part of the system need be revised.

In the rest of this section, the overall organization of SYSPACQ is described briefly. Detailed description of some of the important modules of the program is also included.

After the parsing is completed, the tables generated during parsing can be loaded in the ENTRYPRO database called SYSPAC by using the "LOADER" module of SYSPACQ program. The LOADER also functions as a forward reference resolver for the forward item references in Item Processing Commands. An appropriate error message is generated if any of the forward references are not satisfied. (See section 2.5).

SYSPACQ program can be logically divided into 3 modules, these are: Initialization routines, Parsing and Loader routines. (See Fig. 4) Code generation is done at the time of parsing.

The Parser is the largest and most complex of these modules. The following sections give a brief description of the important aspect of the implementation of the major modules.

In a COBOL program it is possible to have most of the initial values set in Data division. This feature has been used to initialize the variables. To make the program truly re-entrant, we should move all the intialization aspects of the program within procedure division. The COBOL/3000 [1] compiler separates code from data segement and generates an extra code segment to do the initialization, this makes the code re-entrant. However, for implementation of this program on the compilers which do not generate an extra code segment to do the initializing, it will be necessary to do all the initialization in the initialization routine, to make the program code re-entrant (i.e. sharable among many users).

The following group of variables need to be initialized in SYSPACQ:

1. Maximum length or limits of occurrences,
2. Punctuation and delimiting characters,
3. Temporary variable counters, table indexes (pointers) and table items,
4. Record descriptions and record reader items.

Parsing techniques and loader organizations are discussed in the following sections of this chapter.

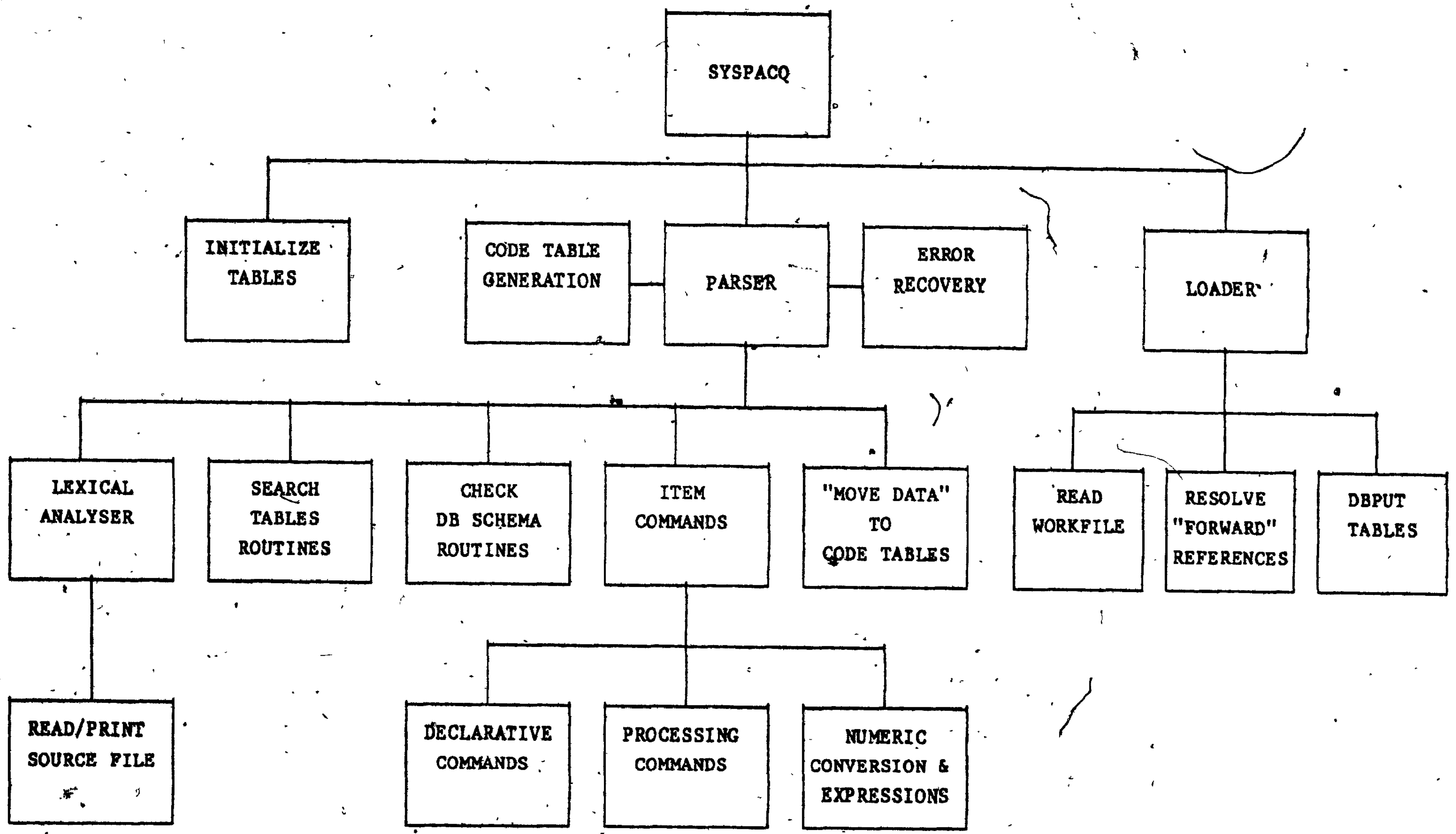


FIG. 4: ORGANIZATION OF SYSPACQ (COMPILER)

2.2 PARSING TECHNIQUES USED AND ERROR RECOVERY

The parsing technique used in this program is sometimes called as LL(1), in other words the program has to look ahead by one symbol only to determine the correct action. Since COBOL permits no recursion, the parser uses sequential algorithm. The program represents the BNF syntax given in Appendix A.

As in the case with every compiler, the most delicate issues arise from error recovery and repair. This program attempts at no repair, however all the errors are flagged below the line and column position where they occur. The current position of pointer is printed along with error code. Appendix "B" contains the listing of error codes generated by SYSPACQ.

The error numbers and organised in a manner that indicate the level at which the error was detected. For example, errors detected by lexical analyser (unknown character etc...) are of the lowest over. On the other hand, errors detected during run time are of the highest level.

The error recovery mechanism is simple in itself. If an error is detected the program discards all the symbols until the next semicolon (;) a sentence delimiter is encountered and the program skips to the next sentence expected in the input string.

The second technique of error recovery is used when parsing parameters of item commands. In this case all symbols until the closing bracket [ )] are discarded and the program skips to parsing instructions which would be found after the closing bracket.

The database schema (DBSCHEMA) errors are flagged right under the item or type which caused the mismatch with the information in DBSCHEMA. The program continues without any error recovery.

2.3 TABLE GENERATION

The table generation in this program should be considered synonymous to code generation of the compiler. The instructions which are used for assembling the contents of the table will be found in lower case letters (see SYSPACQ listing).

The tables themselves are constructed by using the following data structures:

key item	table buffer
16 bytes	240 bytes

Please note that system code is concatinated with all the key items to uniquely identify the key items in the SYSPAC database.

Not all the table buffer space is used. We can categorize all the tables in 3 sets. The first set of the tables are used to describe the user system environment. 4 tables are used for this purpose. For the details of individual table organisation, see section on "Data Structure of Tables". The system title is contained in the first table (or record) and the given string appears as the first line of the form on the screen. Example of a system title:

LESTERS FOODS LTD: A/R SYSTEM

The second set of tables are used to describe the menus in the system. For each line in the menu (i.e. menu item) one table is created. Therefore if a menu has 5 choices then 5 table records will be created. The table of this type contains all the information regarding a particular menu entry description, the users who can evoke it and the action associated with it. The following action can be associated with any menu entry line:

RUN	(R)	Run the Program indicated
STREAM	(S)	Initiate the Job indicated
FORM	(F)	Display the requested Form and execute for processing.
MENU	(M)	Display the requested Menu and accept choice
COMMAND	(C)	Execute the requested operating system command [4]
QUIZ	(Q)	Initiate QUIZ report writing package. [5]

The last set of tables is used for description of the form itself. The following different types of table entries are generated by SYSPACQ:

1. Form Header Table Entry (1 entry)
2. User Option Description Table (1 entry)
3. Lock Description Table Entry (1 entry)
4. Item Description Table Entry (one for each item)

The form header table entry contains the global information about the form, such as dataset or database subforms if any. The user option description table indicates the capability of users with respect to that form, i.e. who can modify, who can inquire etc... The lock descriptor indicates the type of locking strategy that can be used when handling this form. (Not implemented - default locking strategy used).

The most exhaustive of all the tables is the item description table (IDT). There is one table entry for each item in the form. An IDT can be divided into 4 logical parts:

1. Item Definition: gives item name (as in database) the display and storage (DB) type, location of the item in the database buffer and on the screen.

2. Item Flags: These flags indicate Item Declarative and Processing Commands (see section 2.4).

3. Item Description:
   A character string that will be displayed on the screen to tag a field, eg.

```
 CUSTOMER #  |____________|
   (tag)        (field)
```

4. Item Processing Commands: This table contains the coded description of the item processing (editoring, validating) commands (see section 2.7 for data structure details). The item processing commands are described in the following section.

2.4 ITEM PROCESSING COMMANDS

These commands can be divided into two groups: Declarative and Manipulative. Any or all items (in the form) may have one or more of the item processing commands enumerated below. The order in which these commands appear is important and should be maintained when developing the specifications file, although some or all may be omitted. The commands are organised by their frequency of expected usage.

DECLARATIVE COMMANDS	EXPLANATION
1. LOCAL	Declare a local or working item in the form. This item will not be searched in the dataset. The forms designer will have to use one or more item processing commands to set its value, if not an input is used as a field.
2. LIST	Declares a check list item for the form. A subroutine call maybe specified for validation. Check list items are accepted before the form interaction begins and these cannot be modified once accepted. All entries must have same value in the corresponding field.
3. DISPLAY	The current field contents are displayed but cannot be changed by the operator.
4. KEY	An automatic key value generation routine is used; therefore operator may enter a blank key field.

DECLARATIVE COMMANDS	EXPLANATION
5. INIT	This command indicates the intial value of the item when the form is displayed. The default is zero for numeric type items and space for alphanumeric type items. For date type of items, the current system date can be displayed by the following initialization values. Ex: INIT ("$TODAY"), will initialize to sysdate.
6. DEL	The form designer may specify upto 4 characters value which will be inserted into the current field when a deletion is requested, however the dataset entry in not actually deleted, thereby a logical as against a physical deletion is achieved. A form may not contain more than one "deleted flag" flag this is only one DEL command per form. The DEL item must not be a local type of item for obvious reasons. Example: DEL ("OLD")
7. DUPLICATE	A field with this command is set to it's previous contents (duplicated) however operator can change them anytime. This should be the last Item Processing Command for the item.

DATA MANIPULATIVE COMMANDS	EXPLANATION
8. FETCH	This command causes a data fetch from another dataset of master type (i.e. accessed via unique key value). This is performed by the "DBGET" procedure in IMAGE [3]. The key item name should be specified along with the dataset name, database name is optional. See Appendix A for BNF form of the syntax. An error message is displayed if the FETCH command does not succeed for any reason.
9. FIND	This command is similar to FETCH except that no data is "moved". This can be used for validation purposes. For example if a product# is entered then it can be validated against the product master file to ensure that no "invalid" products appear in the database. The execution of the form will not proceed further until a valid value is entered. Out of the two commands FIND or FETCH only one is permitted. Therefore, these two commands are mutually exclusive.

The FETCH example and execution, within its explanation:

```
EXAMPLE: 10,10 "CUSTOMER#" CUST#;
         10,30 "CUSTNAME " NAME ;
                          FETCH (CUST#, MCUSTOMER)
```

Execution of FETCH command is as follows:
Access MCUSTOMER dataset using CUST# as key value and move the data as below.

```
NAME:= NAME.MCUSTOMER
```

DATA MANIPULATIVE

COMMANDS	EXPLANATION
10. CALL	This command initiates a subprogram call to a user specified subprogram. Function calls are not permitted in COBOL.

EXAMPLE: CALL (MOD: I, J, REMAINDER)

Each CALL command has one invisible parameter. It is used for communicating within the ENTRYPRO and the subprogram. The parameter should be defined in the linkage section (COBOL) as the first parameter:

```
01 COMMUNICATION-AREA
   02 CONDITION-WORD    PIC S9(4) COMP.
   02 MESSAGE           PIC X(60).
```

If an error condition is detected during execution of the called subprogram, the subprogram should set the CONDITION-WORD to non-zero (zero value indicates the successful execution), and move an appropriate error meesage in MESSAGE field. The ENTRPRO will display this message to the operator so he or she can correct the data errors.

IMPORTANT: The subroutines are dynamically loaded during execution time and they must reside in the segmented library. Maximum number of parameters in CALL command may not exceed 15.

DATA MANIPULATIVE

COMMANDS	EXPLANATION
11. EVAL	This command evaluates the expression specified with this command and the result is returned into the item to which it belongs. It should contain a valid COBOL type of expression except the square backets [ ] should be used. The constants in the command should not exceed S9(6)V9(3) size. EXAMPLE: OVER-TIME-PAY,EVAL(HOURS*[RATE+0.5]);
12. SHOW	This command causes the contents of the internal buffer to be displayed on the screen at the time of processing an item. This command is useful for the items whose value may depend on the entry of some preceeding items. EXAMPLE: TOTAL, CALL(COMPUTE:AMTS,TOTAL),SHOW; In this example the value returned by the COMPUTE subprogram will be displayed on the screen in the current item field.
13. TOTAL	Maintains a running total using the current item value. Applicable only to detail (sub) forms. The computed total is returned to the specified field. A compare field may also be specified. EXAMPLE: TOTAL (TOTAL-AMT, COMPARE-AMT);

NOTE: The first field of a form is required to be the key item field. The maximum size of key can be 32 characters.

The following Item Processing Commands may be specified with the key item, other commands are ignored:

a) KEY

b) CALL

c) FIND

d) SHOW

In detail form (sub form) the first field must be the same as the header (main) form key item because these forms are linked via this key value. This field may not be modified by the operator. It can be removed from the screen display by setting Row, Col to 0,0 in the field item line in the form. Item Processing Commands on the key field in a detail form are ignored.

2.5 FORWARD REFERENCE RESOLUTION

Only CALL item processing command may have reference to an item name in the form which has not been encountered by the parsing program. For example:

```
10,10 CUST#, CALL(GETADDR: CUST#, NAME, ADDR);
10,30 NAME , TYPE(X:32), DISPLAY;
11,10 ADDR , TYPE(X:60), DISPLAY;
```

In the above example NAME and ADDR occur after command CALL. The parser cannot supply the item#'s until some later stage in parsing. One solution to forward reference is "back patching" but this approach is unsuitable for the implementation in ENTRYPRO parser because the table entry pertaining to a particular item is written to a sequential file as soon as it is completed.

The ENTRYPRO parser (SYSPACQ) generates for each item not yet parsed, a temporary item reference# (reference#'s starting at 300 are of this type) and this is entered into a global table of unresolved item references. Hopefully, when the forward referenced item name is encountered within the form being parsed, the actual item reference# is entered into the table of unresolved references. During the load phase all the unresolved references are replaced by the actual reference# as found in the global table of unresolved references. All unresolved references are displayed on the screen along with an appropriate error message.

Fig. 5 gives the organization of Forward References Table.

2.6 ORGANIZATION OF LOADER MODULE

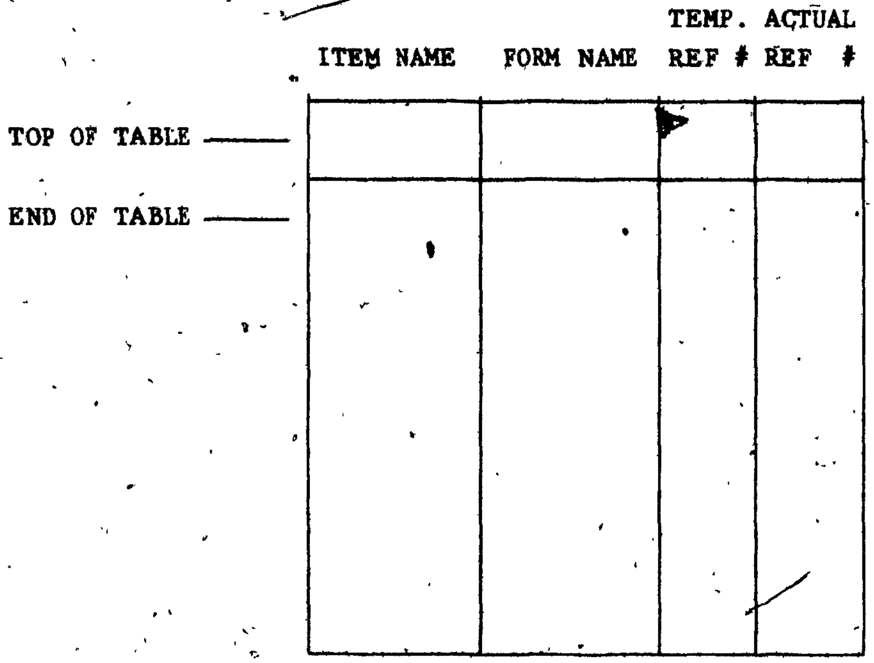
This module reads the work-file containing the tables as generated by the parser. The load file is organised as a simple keyed sequential access file and stored in the SYSPAC database, the organization of SYSPAC is given in FIG. 6. Each menu or form is reduced to several tables as described in DATA STRUCTURES OF CODE TABLES (Section 2.7)

Each table may be represented by one or more records linked to a single key. During the execution this key is used for retrieval of the records (tables). The tables are loaded into SYSPAC-DTL dataset.

This module provides an additional function that satisfies the forward reference items discussed in the previous section. If a table is flagged by the parser as having an unsatisfied forward reference item, the module evokes routines which search the global table of forward references and substitutes the temporary references with the corresponding actual item references found in the table.

This step is performed by this module because the parser is a single pass type. The tables are written into the work-file as they are generated. The loader module reads the work file sequentially and inputs the records into SYSPAC database after satisfying forward references, if any.

FIG. 5: ORGANIZATION OF FORWARD REFERENCES TABLE



```
<<                FIG 6: SYSPAC DATABASE SCHEMA LISTING                >>
<<  -------------------------------------------------------------------  >>
<<  NOTE: THIS DATABASE STORES MENUS & FORMS FOR ENTRYPRO SYSTEM       >>
<<  -------------------------------------------------------------------  >>

$CONTROL BLOCKMAX=1400
BEGIN
   DATA BASE SYSPAC;
   PASSWORDS:  10 READER;
               20 WRITER;

   ITEMS:     MF-NAME,    X20;  <<KEY= SYSNAME (4 BYTES) + FORMNAME  >>
              BUFFER,     X250; <<DATA: BUFFER FOR  ENTRYPRO TABLES  >>

   SETS:
      NAME:  SYSPAC-MASTER, AUTOMATIC;  << DATASET FOR KEYS ONLY     >>
      ENTRY: MF-NAME (1 );
      CAPACITY: 200;

      NAME:  SYSPAC-DTL, DETAIL(10/20); << ENTRYPRO TABLES DATASET >>
      ENTRY: MF-NAME (SYSPAC-MASTER),
             BUFFER;
      CAPACITY: 1000;

END.
<< ------------------------------------------------------------------- >>
```

2.7 DATA STRUCTURE OF CODE TABLES

DATA STRUCTURES SYSTEM ENVIRONMENT DESCRIPTION TABLES

1. SYSTEM TITLE TABLE

X16	X(80)	
$TITLE	system title	not used

2. USER NAME AND USER CLASS NUMBER TABLE

X16	X8	X2		
$USERS	USERNAME	nn		
	1st pair		2nd pair	(maximum of 10 pairs)

3. DATABASE AND PASSWORD TABLE

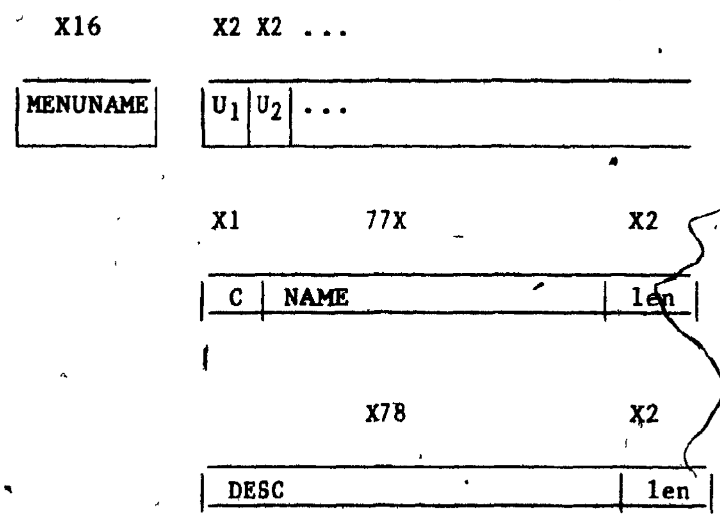
X16	X8	X8		
$BASES	BASENAME	PASSWORD		
	1st pair		2nd pair	(maximum of 10 pairs)

4. FILES AND LOCKWORD TABLE

X16	X8	X8		
$FILES	FILENAME	LOCKWORD		
	1st pair		2nd pair	(maximum of 10 pairs)

NOTE: All the records are organised as: 16 bytes key and 240 bytes of data buffer. The tables are not drawn to scale.

DATA STRUCTURE MENU DESCRIPTION TABLE



Ui :: = User class number.

NAME :: = A Program, Job, Form or Menu name. It may also contain a legal O/S command.

C :: = The command byte is one byte code, which indicates the following actions:

R = Run Program	NAME
S = Initiate Job	NAME
F = Display form	NAME
M = Display Menu	NAME
C = Execute the O/S command contained in	NAME

DESC :: = Menu line description as displayed on the screen.

len :: = Length of the string in bytes.

NOTE: One table entry for each menu line is generated by the parser.

DATA STRUCTURE FORM DESCRIPTION TABLE

Each form is described by one header table, one options table, one lock descriptive table and one or more item descriptive tables (one table for each item).

Form Header Table

X16
FORMNAME

X2	X16	X16	X8	9(4) COMP	X4
FH	DATASET NAME	DATABASE NAME	PASSWORD	N	OPTN

X16	X16	
SUBFORM1-NAME	SUBFORM2-NAME	

Maximum of 5 subforms

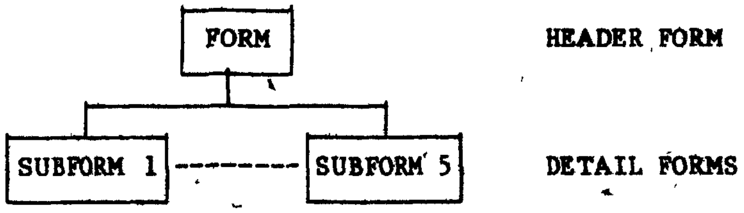
where: N = The maximum number of entries depending on the one unique key in the dataset.

OPTN = There are 4 flags indicating the possible operation on the dataset related to the form.

A (add entry), D (Delete)

M (Modify), I (Inquire)

NOTE: Each form may have up to 5 subforms.



DATA STRUCTURE OPTION DESCRIPTION TABLE

This table defines which user can perform which operation, eg. Add entry by users 10, 11 etc...

X16	X2	X2	X2	X2		
FORM NAME	OP	AD	U_1	U_2		List users allowed to add entries
		DE	U_1	U_2		List of users allowed to delete entries
		MO	U_1	U_2		List of users allowed to modify entries
		IQ	U_1	U_2		List of users allowed to inqure on this dataset.

LOCK OPTION TABLE

The lock descriptor table is supposed to dictate the locking strategy that should be applied during update:

X16	X2	X238
FORMNAME	LD	LOCK DESC

LOCK DESC :: =	READ	-	Exclusive read only
	UPDATE	-	Non-exclusive read but exclusive update
	WRITE	-	exclusive read and update (including adding new entries)

DATA STRUCTURE ITEM DESCRIPTION TABLE: (IDT)

Each item in the form is described by using the following table. The item tables are stored in the order of their occurence in the form from left to right, up and down.

X16	X2	X16	X2	9(4) COMP	9(4) COMP	9(4) COMP	9(4) COMP	9(4) COMP
FORMNAME	REC TYPE	ITEM NAME	DATA TYPE	OFFSET	len	decimal point	row	col

Item Command Flags

X1	X1	X1	X1
local	display	key	other

Display Description

X32	9(2)	9(2)	9(2)
DESC	row	col	length

CPARM buffer

X158
ITEM PROCESSING COMMAND (IPC) PARAMETERS

where:

Itemname	=	Item name as given in DB schema.
Subscript	=	If the item is compound type, gives subscript.
Datatype	=	Item type as given by DB schema.
Offset	=	Relative position of the item within the entry of dataset.
len	=	Display length
decm	=	Decimal position (if applicable)
row	=	Cursor row position.
col	=	Cursor column position.
IPC	=	For details see next page.

NOTE: Item display description (DESC) may be extended upto 79 bytes. The excess bytes are stored in the higher bytes of CPARM buffer.

ITEM PROCESSING COMMANDS PARAMETER BUFFER (CPARM)

The item processing commands list describes the various commands and associated parameters if any. (CPARM byte array will contain this table).

X18	X4	X6	X4	X4	Xn	Xn
INIT	DEL	FETCH	FIND	TOTAL	CALL !	EVAL !

terminator symbol (!)

where:

INIT ::= Defines the intialization value for the given item. Depending on the data type of the item it may contain ASCII string or some numeric quanaity.

DEL :: = At times, the deletion of entries may be symbolic, that is no physical deletion by DBMS is requested but a flag is set to indicate the deletion. Hence DEL is the flag value or string.

FETCH ::= In case the item being described does not belong the primary or main dataset of the form then we use this command to bring data item from the auxiliary or secondary dataset:

FETCH is described as below:

9(2)	9(2)	9(2)
item#	dset#	dbase#

item# ::= DBMS Internal date item#

dset# ::= DBMS Internal dataset #

dbase# ::=Reference to the Database and Password table.

FIND :: = If the value of the item is validated by referencing another dataset containing this item this command should be used. It will help maintain database consistency.

FIND contains the following information.

9(2)	9(2)
dset#	dbase#

For explanation of above non terminal phrases see FETCH command.

TOTAL:: = Item will be used for accumulating totals and the total is returned to the specified item. A compare field may also be specified. Initial value of TOTAL is taken from item # one.

TOTAL contains following parameters:

9(2)	9(2)
item#	item#

CALL :: = A subroutine call may be required for processing or editing the item. This command allows the user to specify the subroutine command. The format of this IPC is:

X16	9(2)	9(2)		
SUBPROGRAM	P1	P2		!

SUBPROGRAM ::= The subprogram or subroutine which is dynamically evoked to do the processing.

Pi ::= The item# within the form which is passed as parameter value. Parameters as string canstants may be also passed.

EVAL :: = The item may be processed by means of an expression, which is evaluated and the result assigned to the item,

eg. AMT = QTY*PRICE.

The items reference in the expression are reduced to item# relative to the form and constants if any are stored as below; the constants may not be larger than 999999.999.

```
     S9(6)V999COMP
|J|                 |
```

The following operators are permitted:

+ addition

- subtraction

* multiply

/ divide

[ ] parameters to alter the operator precedence.

During compilation phase, the syntax of the expression is not checked, instead if there are any errors in the expression, these will be discovered by the interpreter. (See Section 3.5).

3. ENTRYPRO EXECUTIVE PROGRAM

3.1 ORGANIZATION OF EXCUTIVE PROGRAM (SYSPEXEC)

This program interprets the code tables generated by the compiler program (SYSPACQ). These tables are stored in the "SYSPAC" database managed by IMAGE/3000. Accessing of SYSPAC database is transparent to the user and the applications program.

The program organization of SYSPEXEC is given in Fig. 7 [6]. The INIT module initialises the system temporary variable tables. It also accepts the user passwords and validates it against the specified user passwords in the specifications file.

Two main modules, one for MENU HANDLING and one for FORM HANDLING tasks are part of this program. A menu stack is maintained to enable return to the appropriate menu after completion of the tasks.

The last module is used for displaying different types of ERROR messages. It will also terminate SYSPEXEC upon normal or abnormal (error aborts) exits from the program.

A set of screen routines are envoked to perform the various I/O activities on the screen. Similarly database procedures (IMAGE) are called to do the actual data transfers to the databases.

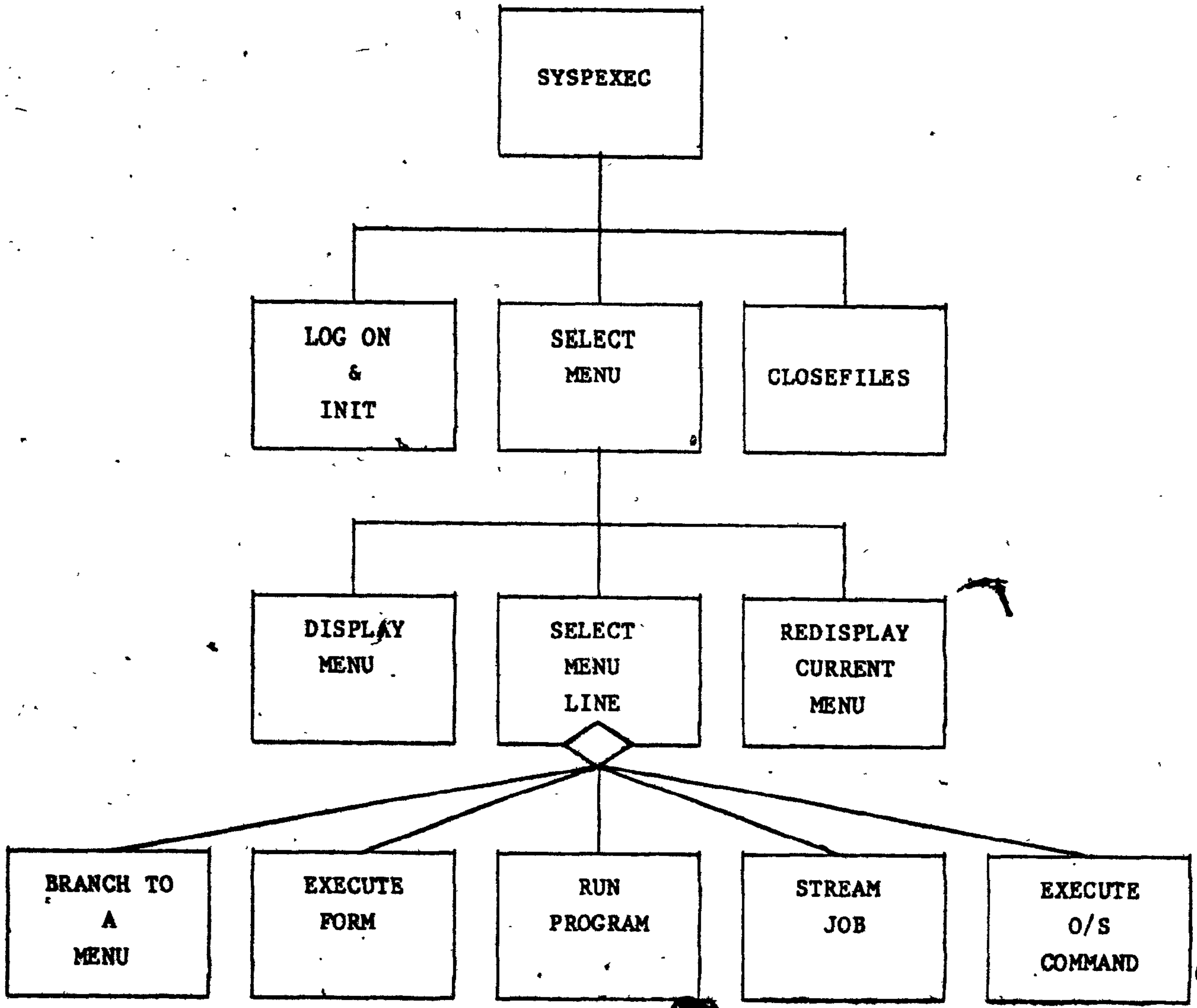


FIG. 7: ORGANIZATION OF SYSPEXEC

3.2 DATA STRUCTURES IN SYSPEXEC PROGRAM

The main type of data structures used in this program are tables, stacks and simple or compound type variables. Since Hewlet Packard COBOL does not allow Real type of variables, binary integers (COMP.) with the assumed decimal point implement the real type. ENTRYPRO can handle numbers up to 4 digits beyond the decimal point.

The important data tables, data item groups and buffers in the program are described below:

SYSPAC-REC:	Buffer to receive tables from SYSPAC database. This buffer is 240 characters. FORM-LINE group gives the data items for the current form field.
IMAGE-FIELDS:	Variables and status array used for communicating with IMAGE/3000 database management system.
DATA-ITEM-LISTS:	Item name lists used by SYSPAC database (also implemented using IMAGE/3000).
USER-TABLE:	Contains user passwords and corresponding user #'s as defined in the specification file.
BASE-TABLE:	Contains the names and passwords of the user defined databases.

DATA STRUCTURES (Cont'd)

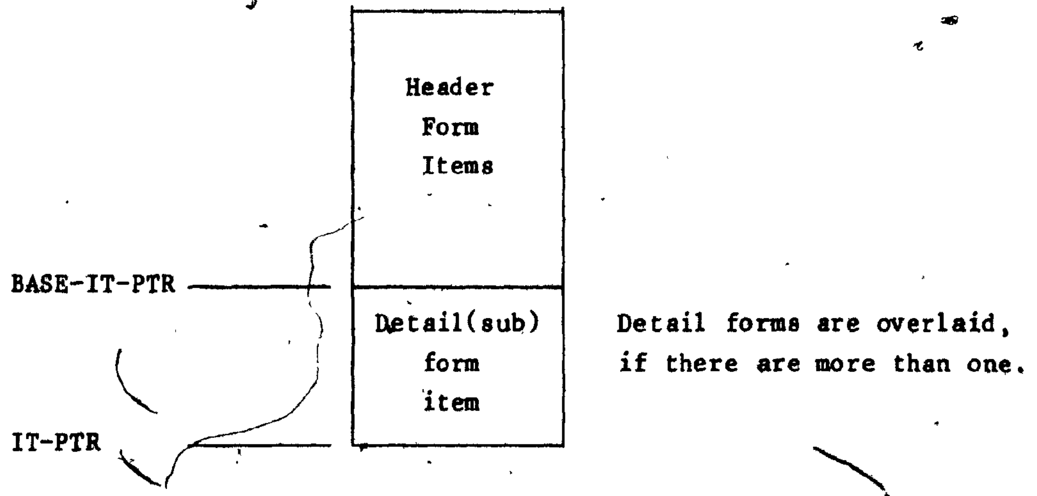
FORM-HDR:	This group of data items define the global information of the current form.
MENU-LINE-TABLE:	A table to story the current menu description and its associated data.
ITEM-TABLE:	It contains the item definition and edit specifications (see Fig. 8).
RETURN-NAME-STACK:	Stacks the names of previous menus (father menu). A maximum depth of 5 menus is allowed.
PNAME-TABLE:	Table of user called subroutine located in the segmenter library.
PCALL-AREA:	Data items used during subroutine loading from segmenter library (SL).
FORM-LIST:	Contains the data items list of the current form, used for calls to IMAGE.
DATE AREA:	This data buffer is used by the current form for receiving and sending data from IMAGE/3000.

DATA STRUCTURES (Cont'd)

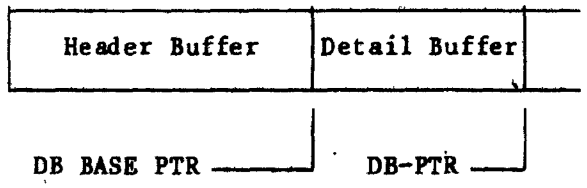
COM-AREA:	Communication area used for user defined subroutines (in SL) and ENTRYPRO.
MESSAGES:	ENTRYPRO messages for interacting with the screen operation.
COMD-MESSAGES:	ENTRYPRO screen directive commands as displayed to the operator.
Miscellaneous:	These are many other temporary variables which are used for screen control, editing and numeric conversion.

FIG. 8: ITEM DEFINITION AND DATA TABLES

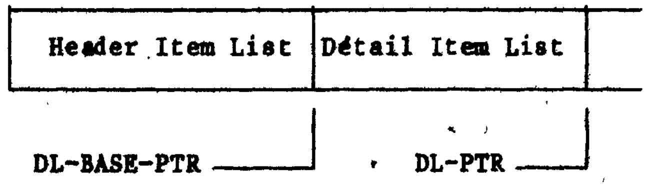
a. Item Table Structure



b. Data buffer and data item list.



c. Data Item List



3.3 MENU DISPLAY FUNCTIONS

During execution of system INIT module the global information such as system title, user passwords database and passwords are read into the system tables. Immediately after this step the "MAIN-MENU" menu is loaded from the SYSPAC database. The menu description lines are displayed and the operator is asked to select one of the lines. If the security class of the user is found in the USER# table for the selected line then the associated command with menu line is executed. With each line only one of the following commands may be associated:

MENU: menu name	= Branch to a son menu given and display it.
FORM: form name	= Display the indicated form and allow operator to manipulate database.
RUN: progname	= Run the selected program. Control returns to ENTRYPRO after termination.
STREAM: jobname	= Initiate the selected job in batch mode and return to the calling menu.
COMMAND: description	=Any HP3000 O/S command may be executed.
QUIZ: quizfile	= Invokes QUIZ report writer using the specified file name.

3.4 FORM DISPLAY FUNCTIONS

System title and form title are displayed on the first and second row of the screen, therefore the actual form description should be started from the third line onwards.

Form display functions consists of outputting the field description at the specified cursor position (row, column). The field descriptions are displayed at the time of reading of the tables from SYSPAC database. All the tables belonging to a form are chained together for easy retrieval. The item definition and edit information is moved into a table established for this purpose (see Section 3.2). The data item names are assembled in a list which is used during the next phase for interacting with the database.
The data buffers used for receiving data from the screen and updating the database are initialized as desired by the optional INIT command. The default initial value for ASCII type fields is spaces and zero for the numeric type fields. Items not referenced in a form are initialized to binary zeros by IMAGE DBMS when creating an entry.

The command prompts and error or action confirmation messages are displayed at the bottom of the screen.

3.6 FORM EXECUTIVE FUNCTIONS

The first field on the screen is assumed to be the key item field. A link in database must exist for this item. The operator is asked to enter the first field and ENTRYPRO initiates a database call to retrieve the corresponding entry in the database. If an entry is found then it is displayed; otherwise it is assumed that the operator wants to add a new entry with the given key value and the operator is prompted to enter values into other fields of the screen.

In the instance where an existing entry has been located, the operator may either modify or delete it. This can be accomplished by either selecting a particular field # or the entire form by selecting "M" option. A "get-field" routine is used for reading (input) the required ASCII characters and converted to the proper type. Item processing and editing commands described below are executed during data entry/modification phase, field by field. If a field is skipped (not entered or modified) then no processing takes place.

To update the database, operator may select "U" option or simply press CR at the end of the screen, when handling a detail form.

Key item may not contain any processing commands except CALL. The item processing commands execution sequence is described in some detail below:

FETCH: This command is used for accessing display information from another dataset which may belong to another database. An appropriate IMAGE call is made with the specified key information and is then displayed.

FIND: This command may be used for validating an item against a database master set entry. Performed through an IMAGE call.

CALL: A call to any user defined procedure is generated. The procedure must be located in the segmenter library (SL.PUB). ENTRYPRO assembles the parameters and makes a branch to the procedure. An implicit parameter is assumed which must be defined in the procedures parameter list. It should be defined as below (COBOL).

```
     01 COM-AREA.
        05 C-CODE    PIC S9(4) COMP.
        05 C-MSG     PIC X(60).
```

At the time of call C-CODE is set to 1 if there is an existing entry in database. The procedure called must set C-CODE to 0 if there are no errors, otherwise set it to a non-zero interger and move the error message to C-MSG. This message will be displayed to the operator. If the current item is DISPLAY type then ENTRYPRO will move the cursor (to accept data) to the first non DISPLAY field prior of the present field, if this field happens to be the first field (key) then screen is cleared and the operator is asked to select another key value.

A procedure call may be specified at the key item level. It is assumed that this call is being used for validating the key value and not editing it.

The maximum number of user defined parameters may not exceed 15, not including the COM-AREA parameter.

The internal processing of CALL command involves the sucessful execution of the following steps:

1. Decode the call parameter list and set PN to the number of parameters.

2. Search PNAME-TABLE. If proc. name is found then set PLABEL(i) to procedure name
   Else move 0 to PLABEL(i) and PIN(i).

3. Load the parameters into P-STACK by calling "LOAD'PARM" routine.

4. Call "LOAD'PROC" routine. This routine will load the procedure (PLABEL) from SL.PUB. This step is skipped if procedure is already loaded (PLABEL = 0).

5. Branch to the procedure and do the processing as required. Program control returns to ENTRYPRO at end of the called procedure processing.

6. This step is executed only once at the end of the form.
   Call "UNLOAD'PROC" using PIN(i).
   Repeat this step for all the procedures.

EVAL: This command is used when specifying an arithmetic expression in item processing. The expression specified is a COBOL type of the expression except (and) are substituted by [and] respectively.

The expression is not parsed during the compile phase because only quite simple type of expressions are expected. Command EVAL is performed by first bringing the expression into postfix form. This is done by considering operator precedence and stacking/unstacking tokens accordingly (see procedure POSTFIX on next page) [7].

Once the expression has been converted to postfix from the evaluation is done by the following algorithm:

```
X: = NEXT TOKEN
WHILE X ≠ E DO
If X = operand then
    PUSH (X)
ELSE (* an operator *)
    POP (T1); POP (T2);
    T: = T1 operator T2;
    PUSH (T)
```

Procedure POSTFIX: Converting Infix to Postfix [7]

```
(* convert the infix expression E to postfix.  Assume the last character of
   E is a "!", which will also be the last character of the postfix.
   Procedure NEXT-TOKEN returns either the next operator, operand or
   delimiter (!).  Character "NULL" with ISP ("NULL") = -1 is used at the
   bottom of the stack.  ISP and ICP are functions which return an
   operator's In-Stack or In-Coming Priority.                              *)

Stack (1):= "NULL"; top:= 1;   (* initialize stack *)
x:= NEXT-TOKEN ( E );
while x ≠ "!" do (* do until end of expression *)
    If x is an operand
       then print (x)
    Else (* x is operator *)
    If x = ")"
       then [while stack (top) ≠ "(" do
                print stack(top); top:= top-1
             END
             top:= top-1]   (* delete "(" *)
       Else (* compare In-Stack Priority with In-Coming Priority *)
            [while ISP (stack(top)) ≥ ICP (x) do
                print (stack(top)); top:= top-1
             END
             Call ADD (x, stack, top)] (* insert x in stack *)
       x:= NEXT-TOKEN ( E )
    END
    while top > 1 do (* Empty the stack *)
      [print (stack (top)); top:= top-1]
END;
```

DEL: This option is applicable only to the master set forms. The indicated item is set to a literal specified with DEL command. When the delete ('D') command code is selected the present entry is flagged. During subsequent accesses the delete flag item if found non empty the entry becomes inaccessible via ENTRYPRO ~~even~~ though the entry exists in the database.

Executive Program Modes of Operation

The ENTRYPRO Executive program operates in either input or command mode. In input mode data in the input fields is received. The operator may switch ENTRYPRO to command code by entering "//". In command mode, ENTRYPRO command codes are accepted. Appendix D gives detail of command codes. The Executive may be switched to input mode by entering "M" command code. ENTRYPRO Executive switches between modes when certain events occur; e.g. end of form.

3.6 SUBFORM (DETAIL) EXECUTIVE FUNCTIONS

A main form (Header Form) may have one or more sub-forms associated with it. A subform (detail form) can have only a one line form. This line is repeated and appended to the previous form. When the screen is full, the top most detail line is deleted and other detail lines are rolled up.

The operator may select any one of the detail forms after "update" of the header form. However if there is only one detail form then it will be displayed automatically. At the end of subform execution, the program returns to the header form.

A list option ("L") is provided with the subforms. It can be used to list all the detail lines in the detail dataset. The detail lines are displayed until screen is full. At this point the list option may be terminated by entering "//" or continued by pressing return key.

Subforms are useful in commercial functions such as invoice description. In a typical invoice there is one header description (eg. customer#, name) but it may contain one or more product items shown on the invoice. A subform is used for manipulating detail lines of the invoice.

Detail lines may be searched by entering the actual line number. The line numbers are automatically displayed by ENTRYPRO. The line number displayed is the actual entry (record) number found in the dataset.

3.7 INITIATING BATCH JOB INTERFACE

If a menu line specifies a STREAM command, then there must be a front end form with the same name as the job file name. This form will be displayed and the operator will be prompted to supply the necessary information. This information will be put into a "mailbox" (if one exists) or appended to the job file. The maximum length of information buffer allowed is 80 characters. After the above action the specified job file is STREAMED (initiated as a batch job) and the current menu is re-displayed. The "mailbox" should be cleared by the receiving program before next "mail" is put into it.

Example: See the sample specification file attached as Appendix E.

3.8 SCREEN DISPLAY/INPUT PROCEDURES

This subsystem consists of number of procedures which perform a simple task such as display a given ASCII string. Briefly, these procedures are displayed below (see program listing for more details); these procedures are written in the System's Programming Language of HP3000. [2]

CLEARSCREEN: Clears the entire screen

CLEARFORM(r): Clears the screen below row(r).

CURSOR(r,c): Positions cursor at a given position on the screen, position (r,c), where r is the row and c is the column.

PRINTEXT (r,c, buffer, l, dx): It displays the given (ASCII) buffer of length l starting at cursor position (r,c) in the display enhancement given by dx character (e.g., half bright, blinking, see attached table).

READNUM (r,c, number, l, k, j, err): Reads a numeric string starting from cursor position (r,c) of length l with jk decimal positions. Error code is returned in err and j contains the actual number of characters entered by the operator.

DISPLAY (buffer, len): Displays the buffer of a given length at the current cursor position, no linefeed at the end of DISPLAY.

INPUTEXT (M, C, buffer, l, j, maxlen): Similar of READNUM but it reads a string of ASCII characters of maximum length given in maxlen, j will contain the actual number of characters entered into buffer. Note that buffer is set to spaces before any data is transferred to it.

SETFIELD (r, c, c2, dx): It is used for setting the field enhancement. The field enhancement will start at position (r,c) and end at position (c2). The type of enhancement is indicated by dx.

LINEDELETE (r,m): It deleted m rows starting at row r.

3.9 DATABASE MANIPULATIVE PROCEDURES

The ENTRYPRO uses following IMAGE/3000 database procedures [3]:

DBOPEN	(base, password, mode, status): It opens the database in a specified mode. Status array always contains the information on the call, set by DBMS.
DBFIND	(base, dataset, 1, status, keyname, value): This call locates the pointer to a given chain in a detail dataset. The key value is given in value parameter.
DBGET	(base, dataset, mode, status, list, buffer, value): It reads an entry from the specified dataset. Following DBGET mode are used in this program:

Mode = 7	A calculated (random) read on master set.
Mode = 5	Read chain forward.
Mode = 6	Read chain backwards.
Mode = 2	Read serially.
Mode = 1	Re-read the current entry.

DBPUT	(base, dataset, 1, status, list, buffer): It adds an entry to the dataset, using the given list and buffer.
DBUPDATE	(base, dataset, 1, status, list, buffer): It updates an entry using the items given by list. The items are in the buffer.

DBDELETE (base, dataset, 1, status): Deletes the current entry from the dataset.

DBLOCK (base, lock description, mode, status): It applies lock (exclusive access) database entries at either entry level or dataset level as specified by lock descriptor. Locks may be gained conditionally or unconditionally (specified by mode).

DBUNLOCK (base, dummy, 1, status): Releases ALL the locks held on a database.

DBEXPLAIN (status): Explains the status information of the last database call.

DBERROR (status, buffer, len): Same as above but a short message is put into buffer.

DBCLOSE (base, dummy, 1, status): This call is used to close a database.

3.10 SECURITY AND ACCESS CONTROL

Security and access control to the system and the databases are implemented with several levels of passwords:

PASSWORD/SECURITY CODE	IMPLEMENTED BY
a) Group Account password	HP3000 System Executive
b) System Code (SYSNAME)	ENTRYPRO Executive
c) User class password	ENTRYPRO Executive
d) Database passwords	IMAGE DBMS

MENU ACCESS

The users are given access to only those menus lines for which their user class password has been validated. Menu lines for other user class (exclusive) are not displayed.

FORM ACCESS OPTIONS

Each user class may be allowed one or more of the following operations on the datasets:

a) Add an entry

b) Delete an entry

c) Modify an entry

User class 99 permits unlimited access to the datasets.

3.11 DATABASE LOCKING AND CONSISTENCY

In a network type of database, the locking is done by the application programs. This can cause delays and starvation to other programs [8]. In ENTRYPRO a locking scheme has been implemented to circumvent this situation. The locking strategy is as follows:

Whenever an existing entry is accessed, its actual contents is saved in a buffer. Subsequently if an update of this entry is requested, ENTRYPRO will lock and re-read the current entry and compare the new buffer values with the saved buffer. If these two buffers have the same contents then data entry is updated. However, if the buffer match fails, no update is attempted and the operator (user) is requested to access the entry once again and the entire sequence of operation is repeated.

While inserting a new entry ENTRYPRO ensures that an entry with the specified key doesn't exist, this is applicable to master datasets only.

Actual locking of the database is done at dataset level. ENTRYPRO attempts to lock the dataset 10 times. If lock cannot be obtained, then a message is displayed. The entry re-read, compare and update functions are performed under locked dataset condition to ensure database consistency. The lock is released after these functions are performed.

When deleting a header type entry all the detail lines (in detail set) are deleted before the header entry is deleted. (Example: Invoice header and detail lines associated with the header). This further ensures data consistency.

BIBLIOGRAPHY

1. COBOL/3000 compiler, Reference Manual
   Hewlett Packard, Santa Clara, Calif., 1979

2. Systems Programming Language, Reference Manual
   ibid

3. IMAGE Data Base Management System, Reference Manual
   ibid

4. Multiprogramming Executive Operating System (MPE III)
   ibid

5. The Complete QUIZ Users' Guide
   Quaser Systems Ltd., Ottawa, 1980

6. Structured Design, by E. Yourdon
   Yourdon Inc., N.Y., 1980

7. Data Structures, by E. Horowitz & S. Sahni
   Computer Science Press Inc., Potomac, Maryland, 1976

8. An Introduction to Database Systems, by C.J. Date
   Addison-Wesley Publishing Co., 1976

APPENDIX "A"

A Form Description & Database Interface

Language (DIL)

BNF syntax (keywords are underlined)

NOTE: i { } means 0 or more times, while [ ] means 0 or 1 time only.

ii Any text enclosed within a pair of (!) is treated as a comment.

```
formsfile ::= SYSNAME:  sysname ; string
              USERS:  username, number   ;
              BASES:  basename, password ;
              FILES:  filename, lockword ;


               menupart
               formpart


              END.
```

NOTE: filename, basename (database name), username are defined as indentifiers.

```
sysname     ::= letter {letter|digit} maximum of 4

identifier  ::=  char  {char}   maximum of 16

menupart    ::= MENU     menuname
                [userno (:) userno] string [MENU menuname |FORM
                          formname | command ]
```

```
formpart ::= FORM: formname , dataset (number) [,basename];
             [DETAIL-FORM: formname  , formname  ;]
             [LOCK-DESC: lockdesc ;]
             [OPTIONS: ADD(number[,number])
                       MOD(number[,number])
                       DEL(number[;number])
                       INQ(number[,number]);]
              line#, col#  string   itemname  [,type]  item commands  ;


              DETAIL: formname, dataset
               line#, col#   string   itemname  [,type]  ,item command  ;



menuname ::=  letter  {letter|digit}
formname ::=  letter  {letter|digit}
dataset  ::=  letter  {letter|digit}
num      ::=  digit {digit}
letter   ::= A|B|......Y|Z|-|#
itemname ::= a valid database item identifier, it may be subscripted or
             an identifier name if it is a local (non database) item.
digit    ::= 0|1|2|......9
lockdesc ::= this should describe the locking strategy which will be
             used when executing this form.  If not specified then the
             default locking strategy is used.
```

```
command      ::= RUN  programname
                 STREAM  filename
                 any other legal command of "MPE" operating system

item command ::= LOCAL|
                 LIST|
                 INIT( string | sign num[.num])|
                 DEL (string|num)|
                 FETCH(item, dataset, database)|
                 TYPE ( type desc )|
                 FIND (dataset, database)|
                 TOTAL (item, item)|
                 CALL (id:parameter  ,parameter |
                 EVAL ( expression )|
                 SHOW |
                 DUPLICATE

string       ::= one or more characters (ASCII) enclosed within single
                 quotation marks (').

paramater    ::= item|constant|string
```

NOTE: If a header form item is referenced in a detail form it should be indicated by $ itemname , so that there is not item/name conflict. Item commands permitted on first field (key field) are KEY, INIT, DISPLAY and CALL.

TYPE: This is the type declarative command. It indicates the edit and display types. Default is same as in database (for local items default is X type and the item length is set to zero).

type desc ::= $\left[\begin{array}{c} \underline{D} \\ \underline{M} \\ \underline{Y} \end{array}\right]$: 6|8

::= $\left[\begin{array}{c} \underline{J} \\ \underline{K} \\ \underline{R} \\ \underline{S} \end{array}\right]$:n[:m]

::= X:n

Where n and m are unsigned integers.

Date Types are:
D = date in DMY format
M = date in MDY format
Y = date in YMD format
The dates can be 6 or 8 bytes in length i.e. with or without embedded "/".

Numeric Types are:
J = number in positive unless specified
K = a positive number (always)
R = a negative number (always)
S = the number is negative by default but operator may specify +

The first unsigned integer gives the total field length (in digits) and the second integer (if specified) indicates the decimal point position.

Character Type
X = string of ASCII characters of a length given by n.

An expression in any legal COBOL expression of arithmetic type of commands but use [,] instead of (,).

expression ::= - expression | item operator item | (expression)

APPENDIX "B"

Error Messages Generated by ENTRYPRO

LEXICAL ERROR#	ERROR MESSAGE
0	Illegal character in the statement.
1	Sysname too long; it should not exceed 4 letters.
2	Identifier too long or incorrectly formulated (identifier should be 16 chars).
3	Integer part of the number exceeds 14 digits; truncation occurs.
4	Real part of the number exceeds 4 digits; truncation occurs.
5	String constant exceeds 80 characters.

SYNTACTIC

ERROR#	ERROR MESSAGE
100	missing punctuation mark: comma (,)
101	missing quote mark (")
102	missing punctuation mark: colon (:)
103	missing opening bracket
104	missing closing bracket or mismatching brackets
105	missing punctuation mark: period (.)
106	missing punctuation mark: semi-colon (;)
107	missing punctuation mark: hyphen (-)
108	NOT USED
119	missing keyword; SYSNAME
120	missing keyword: MENU
121	missing keyword: FORM
122	missing keyword: LOCK
123	missing keyword: OPTION
124	missing keyword: DETAIL
125	missing keyword: REPEAT
126	missing keyword: USERS
127	missing keyword: BASES
128	missing keyword: FILES
129	missing keyword: DETAIL
130	NOT USED

SYNTACTIC

ERROR#	ERROR MESSAGE
200	Expected: MENU, FORM, RUN, STREAM or COMMAND
201	Expected: ADD, DEL, MOD or INQ
202	Expected an item descriptor, eg. LIST, DISPLAY etc...
203	Expected an Integer number (unsigned)
204	Expected an Indentifier name
205	Incomplete specification file or missing END.

SEMANTIC ERROR#	ERROR MESSAGE
300	USERS EXCEED 24; (username, userno) pairs should not be greater than 24.
301	The (Base, Password) pairs should not exceed 10.
302	The (File, lockword) pairs should not exceed 10.
303	Userno not defined in USER:clause.
304	Userno's exceed maximum number of users.
305	Duplicate FORM or MENU name.
306	Menu of Form name reference not found or referenced twice (Look at the menuform table display on the screen for debugging purpose).
307	Corresponding Password or Basename not found.
308	Too many detail form names, should be less than or equal to 10.
309	Error in Detail-Form name descriptor.
310	Error in TYPE declaration (see item type declarative details
311	A local variable must have type declarative preceeding "LOCAL" keyword/or a local variable may not be subscripted
312	Subscript value exceeds that of the data item.
313	Detail form name not declared in "DETAILFORM:" clause of the form.
314	Only numeric type of data may have decimal positioning.

SEMANTIC ERROR#	ERROR MESSAGE
315	OPTIONS: Clause should have one or more of following terminated by semi-colon. ADD(num ,num) [MOD(num ,num)] [DEL(num ,n|m)] [INQ(num ,num)]; use 99 for all users eg. ADD(99)
316	Maximum userno's exceed 24 in "options:" clause (ADD(etc...))
317	The combined length of database buffer and local items connot exceed 600 words (=1200 bytes)
318	Error in Command Parameter declaration.
319	Expected an item-name or a string
320	A COBOL Type of algebraic expression expected.
321	A bad or forward item name referenced in the expression.
322	Too many or too long parameters try to reduce the number of commands on this item.
323	A delete (DEL) item should be of the following type a) The set should be "MASTER" type. b) Image type of the item should be "X" type. c) Item cannot be declared local.
324	Max items in a form including (its detail forms) cannot exceed 80.

APPENDIX "C"

ALLOWABLE LIMITS AND DEFAULT VALUES

1. The identifiers|itemnames should be less than or equal to 16 characters eg. CUSTOMER#, CUSTOMER-NO.

2. The database items should be spelt as in the database schema.

3. Use of subscript is allowed in case the database item is compound type. However, the local items cannot be subscripted.

4. There may be upto 10 user classes only.

5. A menu may not contain more than 20 lines.

6 A form may not contain more than 80 fields, including a subform (one at a time).

7. The string constants should not be more than 78 characters. However, in case of specifying MPE command use only 77 bytes, last byte is CR.

8. The database and user names should not be more than 8 characters. The password and lockwords should not be more than 8 characters either.

9. The largest numbers handled by ENTRYPRO are: Integer 9(14)

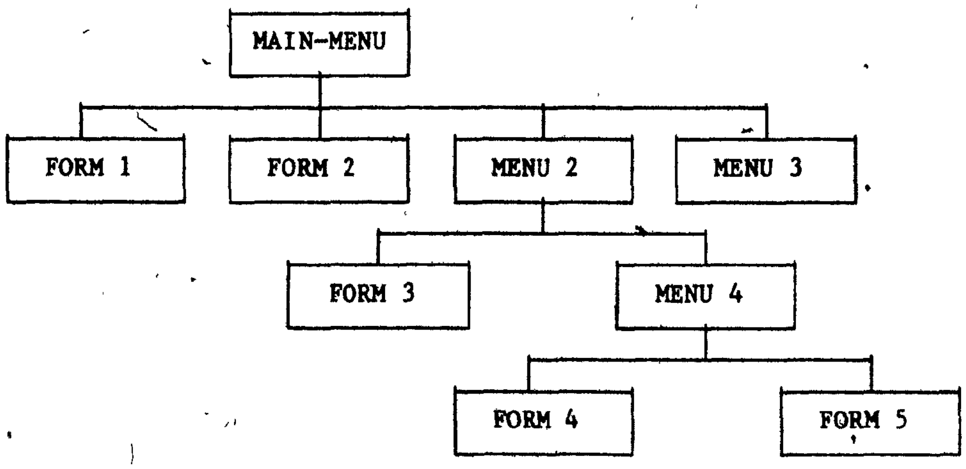
Integer	9(14)
Real	9(14)V9(4)

10. Each keyword such as USERS, BASES, FILES, MENUS, FORM and DETAIL should be started on a new line.

11. Userno's in USERS: clause should be given in ascending order.

    DO NOT use 99 as userno, this number is used by the system to terminate a strip.

12. MENU/FORM reference relationship.



RULES: a) Each menu/form must be referenced by at least one menu.

b) A form cannot reference a menu.

c) Except for MAIN-MENU, all the Forms/Menus must be referenced otherwise they become inaccessible.

d) Maximum depth of menu levels may not exceed 5.

e) Example: MENU/FORM REF TABLE

MENU/FORM NAME	TYPE	REFERENCED BY	FOUND
MAIN-MENU	ME	$HEAD	YES
CUSTOMER	ME	MAIN-MENU	YES
SAMPLE	FO	CUSTOMER	YES
FORM-2	ST	SAMPLE	NO

ME=MENU
FO=FORM
ST=STREAM FILE

13. In any one specification file, the total number of forms and menus must be less than 100.

14. SYSPAC MESSAGE CONVENTION:

    : the system prompt for input of data/command etc...

    ?? (message) An error message.

    (message) A directive for the operator.

    ** (message) A warning or caution for the operator.

    (message) The input string is unacceptable or invalid.

15. A form may have no more than 5 detail forms.

16. The item command parameters total string (characters) length may not exceed 114 characters including keywords, commas and parenthesis:

    eg. GET (name,name,name),CALL(P1,P2...).

17.

Length of DATASETNAMEs	16 chars
Length of DATABASENAMEs	8 chars
Length of PASSWORDs	8 chars
Length of LOCKWORDs	8 chars

18. All local items must be accompanied by type description otherwise the item length is set to 0.

19. Subscripted items of a compound item in a form must appear contiguously.

20. There may not be more than 5 check LIST items in a form. A check LIST item if declared local will not be used for entry validation.

21. There may not be more than 5 TOTAL items in a detail form.

22. There may be upto 80 items in a form. A detail form may contain upto 15 items.

APPENDIX "D"

ENTRYPRO Command Codes

ENTRYPRO executive program (SYSPEXEC) can be in one of the two modes: data input or accept command mode. During data input mode the user is prompted to input data into the fields of the form. ENTRYPRO high lights the field in which input is required. Edit and error messages are displayed at the bottom of the screen.

At the end of the form, ENTRYPRO will automatically switch to command mode however the user may request this switch at anytime during data input by entering "//". While in command mode, a command prompt message is displayed and the user is required to select one of the command mnemonic code. These codes are explained below:

U: Update database with the current form contents (Pressing (CR) will do the same thing).

D: Delete the current entry (on the form). In header forms the detail lines associated with the header entry are also deleted.

M: User wishes to modify the contents of the form. ENTRYPRO is switched to input mode.

nn: (number) In header form it indicates the field# which the user wishes to modify. In detail form it indicates the detail line# which is to be searched and displayed (if found).

L: List the detail lines on the screen.

GO: Initiates a report generation program which is interfaced via the system defined mailbox (MCONTROL).

//: This command has th following semantic significance according to the level in ENTRYPRO:

Field Item : Return to command mode.
Detail Form: Go to header form.
Header Form: Go to calling menu.
Menu Level : Go to previous menu level. If no more levels, then terminate ENTRYPRO.

<<: Move control to previous input field on the screen.

??: Display the current field definition.

APPENDIX "E"

A Sample Specification File and Output

```
HEWLETT-PACKARD 32201A.7.05 EDIT/3000 MON, AUG 17, 1981,  4:38 PM

!JOB SAMPLE,MGR.LESTERS/HP3000,GJAI/Q1980
!COMMENT
!COMMENT     THIS JOB-FILE COMPILES A SAMPLE SPECIFICATION FILE(SYS2)
!COMMENT     THE LISTING  PRODUCED  ARE  ATTACHED  AS  APPENDIX "E".
!COMMENT
!FILE SPECFILE=SAMPLE1
!FILE SALES=SALES.PROD
!RUN BQ.PUB;PARM=2
YES
!EOJ
```

```
:JOB SAMPLE,MGR.LESTERS,GJAI
PRIORITY = DS; INPRI = 8; TIME = UNLIMITED SECONDS
JOB NUMBER = #J43(
MON, AUG 17, 1981,  4:59 PM
HP3000 / MPE III B.01.F2
:COMMENT
:COMMENT    THIS JOB-FILE COMPILES A SAMPLE SPECIFICATION FILE(SYS2)
:COMMENT    THE LISTING  PRODUCED  ARE  ATTACHED  AS  APPENDIX "E".
:COMMENT
:FILE SPECFILE=SAMPLE1
:FILE SALES=SALES.PROD
:RUN BQ.PUB;PARM=2

*** SYSPACQ: COMPILER PROGRAM ENTRYPRO <2> ***


>> SYSPACQ: END OF COMPILE PHASE

>> MENU AND FORM LINKAGE TABLE <<
Menu/Form       Type  Referred.by    # of refs
MAIN-MENU             $HEAD             01
CUSTOMER         ME   MAIN-MENU         01
MCONTROL-FORM    FO   MAIN-MENU         01
PRODUCT          ME   CUSTOMER          01
INVOICE-ENTRY    FO   CUSTOMER          01
JFILENAM         ST   CUSTOMER          01


NO ERRORS IN SPECFILE; WARNINGS= 0000
ELAPSED TIME 00:00:10

DO YOU WISH TO LOAD SYSPAC DATABASE (YES) ?
>> EXISTING VERSION DELETED, ENTRIES=046
>> SYSPAC DATABASE LOADING COMPLETE

END OF PROGRAM
:EOJ
CPU SEC. = 13.  ELAPSED MIN. = 1.  MON, AUG 17, 1981,  4:59 PM
```

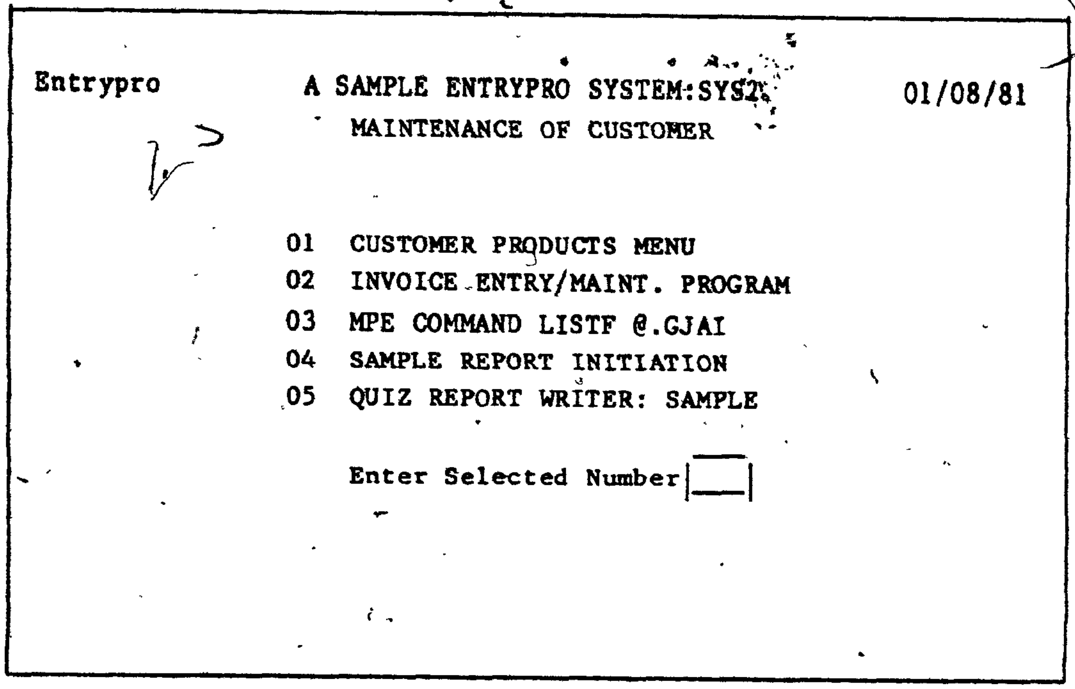
```
ENTRYPRO  VERSION 2.0 (C)              10/16/81   10:01:08


SYSNAME: SYS2;
'A SAMPLE ENTRYPRO SYSTEM: SYS2'

USERS: OPER,  10         ! OPERATOR CLASS PASSWORD          !
       SUPER, 20         ! SUPERVISOR CLASS PASSWORD        !
       SYSMGR,30;        ! SYSTEM MANAGER CLASS PASSWORD    !

BASES: XTEST, WRITER     ! DATABASE AND ACCESS PASSWORD     !
       SALES, READER;

MENU: MAIN-MENU;
      'MAINTENANCE OF CUSTOMER'     MENU CUSTOMER;
20,30 'RUN THE PROGRAM: PROGX999'   RUN PROGX999;
   30 'MAILBOX DATASET MAINT FORM' FORM MCONTROL-FORM;

MENU: CUSTOMER;
      'CUSTOMER PRODUCTS MENU'      MENU PRODUCT;
      'INVOICE ENTRY/MAINT PROGRAM' FORM INVOICE-ENTRY;
20,30 'MPE COMMAND LISTF @.GJAI'    COMMAND 'LISTF @.GJAI';
      'SAMPLE REPORT INITIATION'    STREAM JFILENAM;
      'QUIZ REPORT WRITER: SAMPLE'  QUIZ 'FILE QUIZUSE=F1';

MENU: PRODUCT;
      'RUN PROGX999: SAMPLE PROGRAM' RUN PROGX999;
10,20 'COMMAND:STREAM JOBFILE2.GJAI' COMMAND 'STREAM JOBFILE';

FORM: INVOICE-ENTRY, INVOICE-HDR(1), XTEST;
      DETAILFORM: DTL-FORM1, DTL-FORM2;
      OPTION: ADD(10,20) DEL(10,20) MOD(99); ! 99=ACCESS BY ALL !

      5,10 'INVOICE#   '         INVOICE#;
      6,10 'CUSTOMER#  '   7,10  CUSTOMER#, DUP;
      6,30 'CUST NAME'     7,30  NAME, TYPE(X:32),LOCAL,DISPLAY;
      9,10 'INV STATUS'   10,10  INVOICE-STATUS,TYPE(J:4),INIT(10);
      9,30 'NO BOXES   '  10,30  NO-BOXES, TYPE(K:5:2);
      9,40 'TOTAL AMT  '  10,40  ORDER-AMT, TYPE( J:10:2),DISPLAY;
                                 ! RUNNING TOTAL FROM DTL-FORM1 !
      9,60 'GL TOTAL'     10,60  GL-AMT,TYPE( J:10:2),DISPLAY;
                                 ! RUNNING TOTAL FROM DTL-FORM1 !


      DETAIL: DTL-FORM1,INVOICE-DTL;
      TITLE: 12,2 'LINE# PRODUCT#         QUANTITY         ORDER-AMT'
       0, 0 '  '       INVOICE#, DISPLAY; ! SET BY ENTRYPRO !
      13, 0 '  '       LINE#,TYPE(K:2);
      13, 8 '  '       PRODUCT#;
      13,23 '  '       ORDER-QTY(1),TYPE( J:8 );
      13,38 '  '       AMOUNT,TYPE( J:8:2),TOTAL($ORDER-AMT);

      DETAIL: DTL-FORM2, GL-DTL;
      TITLE: 12,10 ' LINE#        GL-NO           GL-AMT'
       0, 0 '  '       INVOICE#, DISPLAY; ! SET BY ENTRYPRO !
      13, 8 '  '       LINE#, TYPE(K:2);
      13,23 '  '       GL-NO, TYPE(X:8), DUPLICATE;
      13,38 '  '       AMOUNT,TYPE( J:8:2),TOTAL($GL-AMT);
```

```
FORM: MCONTROL-FORM, MCONTROL, XTEST;
      OPTION: ADD(20,30) DEL(99);
      10,10 'PROGKEY        '      PROGKEY;
      12,10 'PROCESS FLAG'         PROCESS-FLG;
      14,10 'CONTROL BUF'          CONTROL-BUF;
      16,10 'DATA BUFR' 17,10      DATA-BUF,TYPE(X:69);

FORM: JFILENAM, MCONTROL, XTEST;
      OPTION: ADD(10,20);
      10,20 'DATE (D/M/Y)'             DATE-IN,  TYPE(D:8), LOCAL,
                                                 INIT('$TODAY');
      12,20 'FIRST FOLIO#'             F-FOLIO,  TYPE(X:4), LOCAL;
      12,45 'LAST FOLIO# '             L-FOLIO,  TYPE(X:4), LOCAL;
      14,20 'REMARKS      '            REMARKS,  TYPE(X:32),LOCAL;

END.
```

```
PAGE  1       HEWLETT-PACKARD 32215B.02.00 IMAGE/3000: DBSCHEMA

<<  SCHEMA OF 'XTEST' DATABASE; USED BY SYSTEM: SYS2  >>
<<  --------------------------------------------------  >>

BEGIN
    DATA BASE    XTEST;
    PASSWORDS:   10 READER;
                 20 WRITER;

    <<                                                    >>
    << THE DATA TYPES IN IMAGE SCHEMA ARE AS FOLLOWS:     >>
    <<       Xn:  n ASCII CHARACTERS                      >>
    <<       J1:  A 16 BIT INTEGER NUMBER (2 BYTES)       >>
    <<       J2:  A 32 BIT INTEGER NUMBER (4 BYTES)       >>
    <<       Zn:  n ZONED DECIMAL DIGITS ( 1 PER BYTE )   >>
    <<       Pn:  n PACKED DECIMAL DIGITS (2 PER BYTE )   >>
    <<                                                    >>

    ITEMS:     <<NAME>>          <<DATA TYPE>>
               AMOUNT,              Z12;
               CONTROL-BUF,         X20;
               CUSTOMER#,           X12;
               DATA-BUF,            X80;
               GL-AMT,               J2;
               GL-NO,                J2;
               INVOICE#  ,           X8;
               INVOICE-STATUS,       J1;
               LINE# ,               X2;
               NO-BOXES,             P4;
               ORDER-AMT,            J2;
               ORDER-QTY,           4J2; << A COMPOUND ITEM>>
               PROCESS-FLG,          X2;
               PRODUCT#,             X4;
               PROGKEY,              X8;

   SETS:       <<DATASET>>   << SET TYPE & SECURITY     >>

      NAME:    INV-MASTER,   AUTOMATIC(10/20); <<MASTER>>
      ENTRY:   INVOICE# (3);
      CAPACITY:  10;

      NAME:    INVOICE-HDR,  DETAIL( 10/20 );
      ENTRY:   INVOICE# (INV-MASTER),
               CUSTOMER#,
               INVOICE-STATUS,
               NO-BOXES,
               ORDER-AMT,
               GL-AMT;
      CAPACITY:  10;

      NAME:    INVOICE-DTL,  DETAIL( 10/20 );
      ENTRY:   LINE#,
               INVOICE# (INV-MASTER(LINE#)),
               PRODUCT#,
               ORDER-QTY,
               AMOUNT;
      CAPACITY:  20;
```

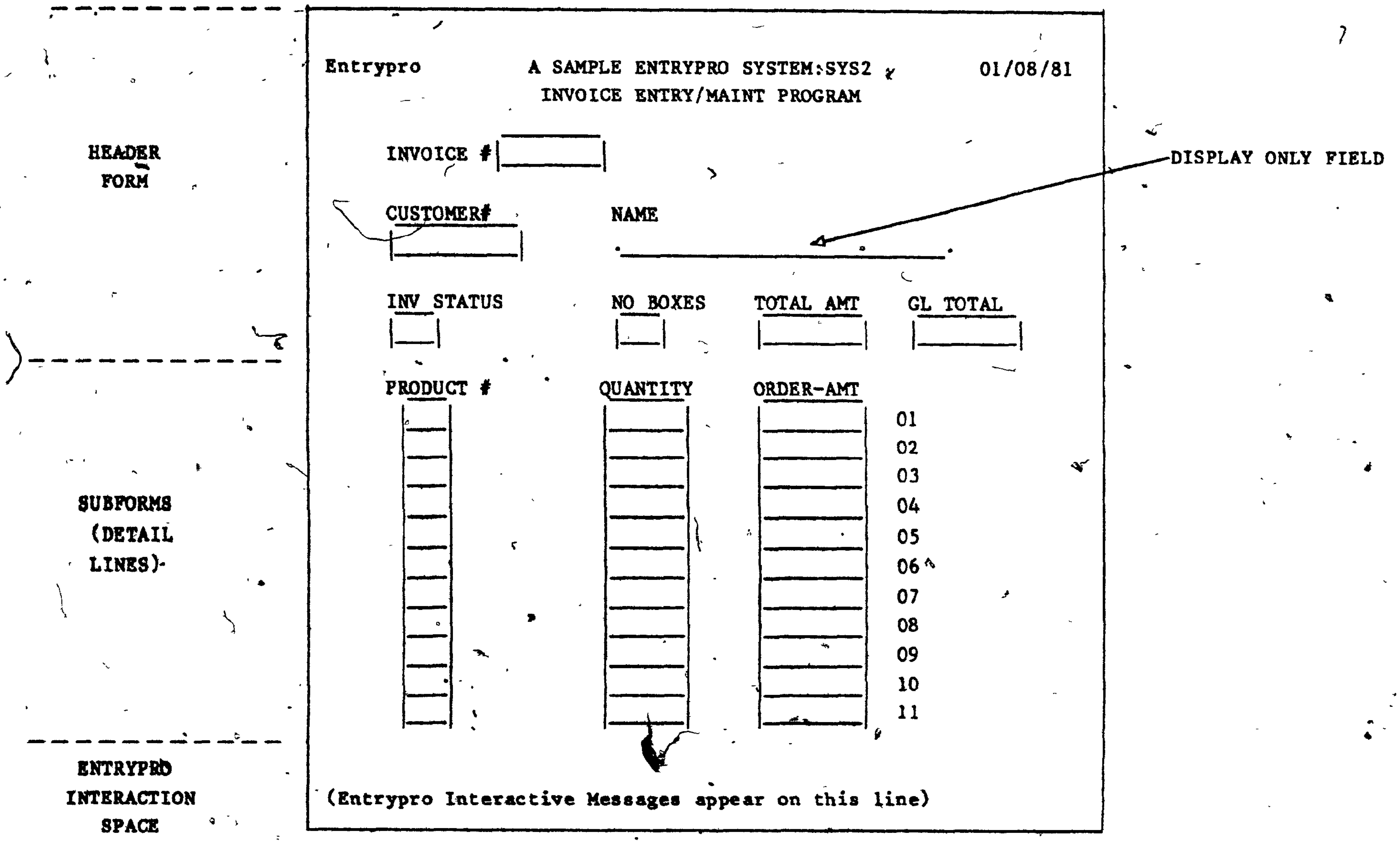
```
PAGE 2        XTEST


        NAME:  GL-DTL,          DETAIL( 10/20 );
        ENTRY: LINE#,
               INVOICE# (INV-MASTER),
               GL-NO,
               AMOUNT;
        CAPACITY: 20;


        NAME:  MCONTROL,        MANUAL( 10/20 ); <<MASTER>>
        ENTRY: PROGKEY( 0 ),
               PROCESS-FLG,
               CONTROL-BUF,
               DATA-BUF;
        CAPACITY: 10;

    END.

   DATA SET        TYPE  FLD  PT  ENTR  MED  CAPACITY  BLK  BLK   DISC
    NAME                 CNT  CT  LGTH  REC            FAC  LGTH  SPACE

  INV-MASTER        A    1    3   4     24    10       10   241   4
  INVOICE-HDR       D    6    1   16    20    10       10   201   4
  INVOICE-DTL       D    5    1   21    25    20       20   502   8
  GL-DTL            D    4    1   13    17    28       14   239   6
  MCONTROL          M    4    0   55    60    10       6    361   9
                               TOTAL DISC SECTORS INCLUDING ROOT:  42


  NUMBER OF ERROR MESSAGES: 0
  ITEM NAME COUNT: 15      DATA SET COUNT: 5
  ROOT LENGTH: 560     BUFFER LENGTH: 502     TRAILER LENGTH: 256

  ROOT FILE XTEST  CREATED.
```

A SAMPLE DISPLAY: SCREEN LAYOUT OF MENU "CUSTOMER"



```
Entrypro         A SAMPLE ENTRYPRO SYSTEM:SYS2         01/08/81
                   MAINTENANCE OF CUSTOMER

             01  CUSTOMER PRODUCTS MENU
             02  INVOICE ENTRY/MAINT. PROGRAM
             03  MPE COMMAND LISTF @.GJAI
             04  SAMPLE REPORT INITIATION
             05  QUIZ REPORT WRITER: SAMPLE

                 Enter Selected Number|__|
```

A SAMPLE DISPLAY: SCREEN LAYOUT OF FORM "INVOICE-ENTRY"



```
Entrypro          A SAMPLE ENTRYPRO SYSTEM:SYS2          01/08/81
                    INVOICE ENTRY/MAINT PROGRAM

     INVOICE #|________|

     CUSTOMER#        NAME
     |________|       ____________________________

     INV STATUS       NO BOXES    TOTAL AMT    GL TOTAL
     |__|             |__|        |________|   |________|

     PRODUCT #        QUANTITY    ORDER-AMT
     |___|            |_____|     |_______|   01
     |___|            |_____|     |_______|   02
     |___|            |_____|     |_______|   03
     |___|            |_____|     |_______|   04
     |___|            |_____|     |_______|   05
     |___|            |_____|     |_______|   06
     |___|            |_____|     |_______|   07
     |___|            |_____|     |_______|   08
     |___|            |_____|     |_______|   09
     |___|            |_____|     |_______|   10
     |___|            |_____|     |_______|   11

(Entrypro Interactive Messages appear on this line)
```